Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 17 (2013) 441 – 448

Procedia Computer Science

Information Technology and Quantitative Management (ITQM2013)

# Toward A Hybrid Approach of Primitive Cognitive Network Process and Particle Swarm Optimization Neural Network for Forecasting

Guangjin Zhang, Kevin Kam Fung Yuen*

*Department of Computer Science and Software Engineering,Xi'an Jiaotong-Liverpool University,Suzhou, China*

**Abstract**

Forecasting by artificial neural network is a popular approach in recent years. This paper proposes a new hybrid approach PCNP-PSONN which combines the Primitive Cognitive Network Process (PCNP) and Particle Swarm Optimization Neural Network (PSONN) for forecasting. PCNP is a rectified approach of the Analytic Hierarchy Process (AHP), to quantify the influence of factors, whilst Particle Swarm Optimization has been used for optimizing the neural network by improving the learning efficiency. The combination of PCNP and PSONN, PCNP-PSONN, can increase accuracy of network through selection of high influenced factors.



Selection and peer-review under responsibility of the organizers of the 2013 International Conference on Information Technology and Quantitative Management

*Keywords: Forecasting; Primitive Cognitive Network Process; Particle Swarm Optimization Neural Network*

## 1. Introduction

Forecasting is the process of predicting the result by some known factors. Back Propagation Neural Network (BPNN) which is the classical approach of Artificial Neural Network (ANN) originally introduced by Bryson and Ho in 1969 [1], is widely used in forecasting area. It is one kind of feed forward neural network which uses gradient descent in error back propagation to reduce the error rate. The error rate is easily to fall into local minimum instead of global minimum value which is the optimal solution in training process by applying gradient descent [3][14]. Whilst the large number of influence factors for forecasting issues may reduce the accuracy of the neural network [3][14]. In this paper, a new hybrid approach PCNP-PSONN has been proposed to select the best factors from data source. An evolutionary computation technique, Particle Swarm Optimization (PSO) has been used for changing the gradient descent for training process to improve the performance. Primitive Cognitive Network Process (PCNP) which is a kind of decision process is introduced for quantifying the subjective judgments of each influence factors to reduce the input number for neural network analysis.

PSO has a strong ability to find global optimistic result. It is used in data clustering and function optimization since it had been developed by Kennedy and Eberhart in 1995 [11]. The basic procedure is to represent every possible solution as particle. All particles have fitness values which are calculated by fitness function and velocities

* Corresponding author. Tel.: +86-(0)512 - 88161517
*E-mail address*: kevinkf.yuen@gmail.com; kevin.yuen@xjtlu.edu.cn.


Selection and peer-review under responsibility of the organizers of the 2013 International Conference on Information Technology and Quantitative Management
doi:10.1016/j.procs.2013.05.057

which direct the flight of each particle. Every particle accelerates in the direction of its own best position and the global best position in the moment of this iteration process [4] [15]. The particle may flow the current optimum particle by updating its position and velocity. The global optimal solution could be generated by the iterations of updating generation [10]. In PSONN, the weight and bias of the ANN are regarded as the particle position for the PSO process, and the error function is regarded as the fitness function. The global optimal pair of weight and bias for neural network is output by PSO algorithm [2].

The primitive cognitive network process (PCNP) [7] [8] [9] is the recent method for computing the influence weight. It is one kind of cognitive network process (CNP). CNP is an approach rectifying the mathematical representation problem of the perception of the difference of pairwise comparisons in Analytic Hierarchy Process (AHP), which is a popular tool for complex decision making problems and has been developed by Saaty in 1980 [13]. [5] proposed a hybrid approach of AHP and BPNN to optimize the classical BPNN. As PCNP is improvement of AHP and PSONN is improvement of BPNN, the combination of PCNP and PSONN is feasible. This study aims to design a PCNP-PSONN model based on combining the primitive cognitive network process (PCNP) and Particle Swarm Optimization Neural Network (PSONN).

## 2. PCNP-PSONN

PCNP is used to quantify the weights of the factors as the input variables of PSONN. The factor of less weight is not selected from the data source. PSONN uses the best influenced factors which are selected form the data source as input variables for network training. The structure diagram is shown in Fig 1 and the algorithm is presented as below:

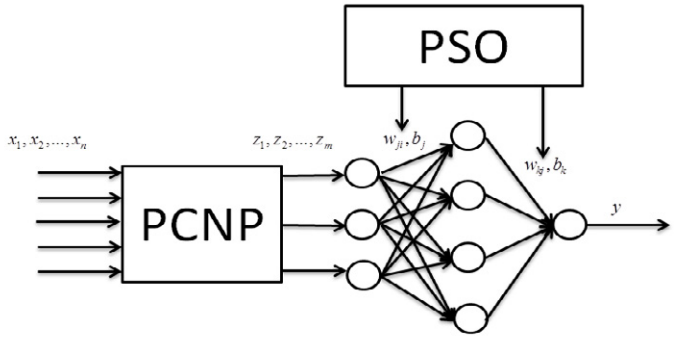


Fig.1 Overview of PCNP-PSONN

### Step 1 Weight determination by Primitive Cognitive Network Process

Primitive Cognitive Network Process (PCNP) [7] [8] [9] is used to determine the weights of factors in PCNP-PSONN. To formulate the problem, the factors are regarded as the criteria in PCNP.

Let the factors of data source be $x_1, x_2, ..., x_n$. The measurement scale schema $\left(\aleph, \overline{X}\right)$ is used to represent the paired comparison between two factors. $\aleph$ is the set of linguistic labels of the paired interval scales such as Equally; Weakly; Moderately; Moderately plus; Strongly; Strong Plus; Very Strongly; Very, very strongly; Extremely. $\overline{X}$ is the numerical representation of $\aleph$, and has form (1) .

$$\overline{X} = \left\{ \alpha_i = {i\kappa}/{\tau} \mid \forall i \in \{-\tau, \ldots, -1, 0, 1, \ldots, \tau\}, \kappa > 0 \right\} \quad (1)$$

Pairwise Opposite Matrix (POM) B which is formed by subjective judgments of decision makers determines the importance of each factor by using form (2) where $v_i$ means the importance of factor $x_i$ . $b_{ij}$ is the different

importance between factor $x_i$ and $x_j$. For instance, $b_{12}=1$ means the factor $x_1$ is weakly more important than factor $x_2$.

$$B=\left[b_{ij}\right]=\begin{bmatrix} 0 & b_{12} & \cdots & b_{1n} \\ b_{21} & 0 & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & 0 \end{bmatrix}\cong\begin{bmatrix} 0 & v_1-v_2 & \cdots & v_1-v_n \\ v_2-v_1 & 0 & \cdots & v_2-v_n \\ \vdots & \vdots & \ddots & \vdots \\ v_n-v_1 & v_n-v_2 & \cdots & 0 \end{bmatrix} \tag{2}$$

$B$ needs to be validated by the Accordance Index (AI) with form (3). If $AI=0$, $B$ is perfectly accordant; If $0<AI\le 0.1$, $B$ is satisfactory; and if $AI>0.1$, $B$ is unsatisfactory.

$$AI=\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}d_{ij}\text{, where }d_{ij}=\sqrt{Mean\left(\left(\frac{1}{\kappa}\left(B_i+B_j^T-b_{ij}\right)\right)^2\right)},\ i,j\in(1,\ldots,n) \tag{3}$$

$B$ is prioritized and normalized to the priority vector $W$ by the cognitive prioritization operator in form (4) and normalization function in form (5) receptively.

$$v_i=\left(\frac{1}{n}\sum_{j=1}^{n}b_{ij}\right)+\kappa \tag{4}$$

$$W=\left\{\frac{v_i}{n\kappa},\forall i\in\{1,\ldots,n\}\right\},\ \sum_{i=1}^{n}v_i=n\kappa \tag{5}$$

The weights will be used in step 2.

**Step 2 Factors selection based on influence weight**

The factor with less weight should not be selected from the data source in this step. The selection rule is shown by form (6).

$$\text{If }W(x_i)<\lambda/n\ ,\ x_n\leftarrow \text{Re}move(x_i)\ ,\ \lambda\in(0,1]\ ,i=1,2,...,n \tag{6}$$

The parameter $\lambda$ is determined by the expert and the value of $\lambda$ should be set to be 1 by default. $n$ is the number of original factors and $m$ is the number of selected factors . If the weight of factor $x_i$ is less than the threshold $\lambda/n$, it should not be selected from the data source. The selected factors $z_1,z_2,...,z_m$ construct into the input variables of next stage.

**Step 3 Weight and bias calculation by Particle Swarm Optimization Neural Network**

PSONN algorithm can be referred to [2] [6] [10].

To initialize the neural network, the number of neurons in three layers should be determined by the size of input data source $z_1,z_2,...,z_m$. In the network, weights $w_{ji}$ and bias $b_j$ of set $W1=(\{w_{ji}\},\{b_j\})$ are in hidden layer, whilst $w_{kj}$ and $b_k$ of set $W2=(\{w_{kj}\},\{b_k\})$ are in output layer. The number of neurons in input layer, hidden layer and output layer are $m$ , $l$ and $o$ respectively, where $i=1,2,...,m$ , $j=1,2,...,l$ and $k=1,2,...,o$. The dimension for searching space $D$ is of the form, $D=m*l+l+l*o+o$, and $d=1,2,...,D$. The number of particles $Q$ (the size of population) is determined by the dimension. Generally, $Q$ is range from 20 to 50 [12], and $q=1,2,...,Q$. The velocity vector $V_q=(V_{q1},V_{q2},...V_{qd}...,V_{q,D-1},V_{qD})$ and position vector $X_q=(X_{q1},X_{q2},...X_{qd}...,X_{q,D-1},X_{qD})$ of $q$ th particle are randomly initialized by form (7). Position vector $X_q$ is composed of $W1$ and $W2$ . The random value of $rands()$ is from -1 to 1. The maximum element of velocity $V_{\max}$ , and the maximum element of position $X_{\max}$ are determined by user. A Particle has position and velocity, and Velocity is used to adjust the position.

$$X_q = \bigcup(W1, W2) = \bigcup\left(\{w_{ji}\}, \{b_j\}, \{w_{kj}\}, \{b_k\}\right)$$
$$V_{qd} = V_{\max} * rands(), X_{qd} = X_{\max} * rands(), V_{qd} \in [-V_{\max}, V_{\max}], X_{qd} \in [-X_{\max}, X_{\max}] \quad (7)$$

The Mean Square Error (MSE) $E$ (form (8)) is regarded as the fitness value of the position $X_q$. For each position $X_q$, $t_{ks}$ is the $k$ th target output of the $s$ th training sample and $y_{ks}$ is the $k$ th forecasting output of the $s$ th training sample. $e_s$ is the MSE of the $s$ th training sample. $E$ is the average MSE of all training samples. $N$ is the number of the training samples and member of sample is defined as $s = 1, 2, ..., N$. In form (9), $z_{is}$ is the input variable of factor $z_i$ which is selected from $x_1, x_2, ..., x_n$ of the $s$ th training sample. In form (10), $f(h_{js})$ is the excitation function which $h_{js}$ means the input value of the $j$ th hidden neuron of the $s$ th sample.

$$E = \frac{1}{N}\sum_{s=1}^{N} e_s, e_s = \frac{1}{2}\sum_{k=1}^{o}(t_{ks} - y_{ks})^2 \quad (8)$$

$$y_{ks} = \sum_{j=1}^{l} w_{kj} f(\sum_{i=1}^{m} w_{ji} z_{is} - b_j) - b_k \ , k = 1, 2, ..., o, s = 1, 2, ..., N \quad (9)$$

$$f(h_{js}) = 1/(1 + e^{-h_{js}}), h_{js} = \sum_{i=1}^{m} w_{ji} z_{is} - b_j, j = 1, 2, ..., l, s = 1, 2, ..., N \quad (10)$$

The individual best position vector $P_q = (P_{q1}, P_{q2}, ...P_{qd} ..., P_{q,D-1}, P_{qD})$ of $q$ th particle and global best position vector $GP = (GP_1, GP_2, ...GP_d ..., GP_{D-1}, GP_D)$ are updated by comparing the fitness value of each particle. If the fitness value of current position vector $P_{current}$ of the particle is less than the one of $P_q$, the $P_{current}$ will replace $P_q$. If the fitness value of current position vector $P_{current}$ of the particle is less than the one of $GP$, the $P_{current}$ will replace the $GP$. The velocity $V_{qd}$ and position $X_{qd}$ are updated by form (11) and (12). If the velocity $V_{qd}$ is larger than the maximum value $V_{\max}$, it will be changed to $V_{\max}$. If the velocity $V_{qd}$ is less than the minimum value $V_{\min}$, it will be changed to $V_{\min}$. If the position $X_{qd}$ is larger than the maximum value $X_{\max}$, it will be changed to $X_{\max}$. If the position $X_{qd}$ is less than the minimum value $X_{\min}$, it will be changed to $X_{\min}$. Parameters $C_1, C_2$ are set to 2.0 typically [6]. They reflect the weighting of acceleration terms which pull the current particle toward the $P_q$ and $GP$. $r_1$ and $r_2$ are random numbers from 0 to 1. Inertia weight $W(k)$ is computed by form (13), where $k$ is current iteration time and $K$ is whole iteration time.

$$V_{qd}(k+1) = W(k)V_{qd}(k) + C_1 r_1 (P_{qd} - X_{qd}) + C_2 r_2 (GP_d - X_{qd}) \quad (11)$$

$$X_{qd}(k+1) = X_{qd}(k) + V_{qd}(k+1) \quad (12)$$

$$W(k) = W_{\max} - (W_{\max} - W_{\min})(k / K) \quad (13)$$

The procedure is iteratively calculated until it reaches the iteration time or $E \le \varepsilon$. $\varepsilon$ is the expected precision. The global best position of final iteration is the optimal solution. The elements of the global best position are the optimal parameters of weight and bias for the PSONN. The weight and bias are used to complete the network for training. The forecasting output with test sample is calculated by form (9) and (10). Furthermore, the trained neural network can test performance with comparing the target output and forecasting output of test sample by error form (8).

## 3. Simulation

The aim of this instance is to train the network with training samples and use the trained neural network to

forecast the output value with testing samples by PCNP-PSONN. There are five samples $\alpha_1,\ldots,\alpha_5$ which contain the input variables of eight factors $B_1,\ldots,B_8$ and their target outputs are $t_1,\ldots,t_5$. Four samples $\alpha_1,\alpha_2,\alpha_3,\alpha_4$ are used to train the network and the sample $\alpha_5$ is used to test the network. The data source is shown in Table 1:

Table 1: The data source

| Factor | Train | | | | Test |
|---|---|---|---|---|---|
| | $\alpha_1$ | $\alpha_2$ | $\alpha_3$ | $\alpha_4$ | $\alpha_5$ |
| $B_1$ | 1 | 3 | 5 | 7 | 9 |
| $B_2$ | 1 | 2 | 3 | 4 | 5 |
| $B_3$ | 2 | 3 | 4 | 5 | 6 |
| $B_4$ | 3 | 4 | 5 | 6 | 7 |
| $B_5$ | 2 | 1 | 2 | 1 | 2 |
| $B_6$ | 1 | 2 | 1 | 2 | 1 |
| $B_7$ | 4 | 5 | 6 | 7 | 8 |
| $B_8$ | 0 | 1 | 2 | 3 | 4 |
| $t_s$ | 2 | 4 | 6 | 8 | 10 |

**Step 1 Weight determination by Primitive Cognitive Network Process**

The pairwise opposite matrix and importance determined by form (1) and (2) are shown in Table 2. The Accordance Index is computed by form (3) and weights are computed by form (4) and (5). The AI of the POM is equal to 0, which means the POM is perfectly accordant.

Table 2: Pairwise Opposite Matrix and their weight (AI=0)

| | $B_1$ | $B_2$ | $B_3$ | $B_4$ | $B_5$ | $B_6$ | $B_7$ | $B_8$ | W |
|---|---|---|---|---|---|---|---|---|---|
| $B_1$ | 0 | 1 | 2 | 2 | 8 | 7 | 2 | 3 | 0.174 |
| $B_2$ | -1 | 0 | 1 | 1 | 7 | 6 | 1 | 2 | 0.158 |
| $B_3$ | -2 | -1 | 0 | 0 | 6 | 5 | 0 | 1 | 0.143 |
| $B_4$ | -2 | -1 | 0 | 0 | 6 | 5 | 0 | 1 | 0.143 |
| $B_5$ | -8 | -7 | -6 | -6 | 0 | -1 | -6 | -5 | 0.049 |
| $B_6$ | -7 | -6 | -5 | -5 | 1 | 0 | -5 | -4 | 0.064 |
| $B_7$ | -2 | -1 | 0 | 0 | 6 | 5 | 0 | 1 | 0.143 |
| $B_8$ | -3 | -2 | -1 | -1 | 5 | 4 | -1 | 0 | 0.127 |

**Step 2 Factors selection based on influence weight**

The input variables of factor $B_5$ and $B_6$ are removed from the data source, as their weights 0.049 and 0.064 are less than $\lambda / n = 1/8 = 0.125$ by form (6). The six factors $B_1, B_2, B_3, B_4, B_7, B_8$ are selected and shown in Table 3.

Table 3: The selected data

| Factor | Train | | | | Test |
|---|---|---|---|---|---|
| | $\alpha_1$ | $\alpha_2$ | $\alpha_3$ | $\alpha_4$ | $\alpha_5$ |

| | | | | | |
|---|---|---|---|---|---|
| $B_1$ | 1 | 3 | 5 | 7 | 9 |
| $B_2$ | 1 | 2 | 3 | 4 | 5 |
| $B_3$ | 2 | 3 | 4 | 5 | 6 |
| $B_4$ | 3 | 4 | 5 | 6 | 7 |
| $B_7$ | 4 | 5 | 6 | 7 | 8 |
| $B_8$ | 0 | 1 | 2 | 3 | 4 |
| $t_s$ | 2 | 4 | 6 | 8 | 10 |

**Step 3 Weight and bias calculation by Particle Swarm Optimization Neural Network**

In Table 3, the input variables of factors $B_1, B_2, B_3, B_4, B_7, B_8$ of sample $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ and the target output $t_1, t_2, t_3, t_4$ of sample $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are used to train the network. The number of neurons in input layer, hidden layer and output layer are 6, 6 and 1 respectively, i.e. $m = 6, l = 6, o = 1$. The dimension is $D = 6*6+6+6*1+1 = 49$. The number of particles is $Q = 20$. The velocity of $V_{qd} = 0.1$ and position of $X_{qd} = 1$ for $q$ th particle are randomly initialized by form (7). The range of velocity $V_{qd}$ is from -1 to 1 and the range of position $X_{qd}$ is from -3 to 3. The weight and bias are $w_{ji} = 1, b_j = 1, w_{kj} = 1, b_k = 1$ . In addition, $C_1 = 2, C_2 = 2$ and $r_1 = 0.5, r_2 = 0.5$ . Inertia weight $W(k)$ is from 0.4 to 0.9. The whole iteration time is $K = 30$ and the expect precision is $\varepsilon = 10^{-5}$ .

The output of $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ is calculated to achieve fitness value of current position $P_{current}$ which is the position $X_1$ of the particle $\beta_1$ by forms (8) - (10). $\beta_q = \beta_1, \beta_2, ..., \beta_{20}$ is a particle.

$$
\begin{aligned}
y_{11} &= \sum_{j=1}^{l} w_{1j} f(\sum_{i=1}^{m} w_{ji} z_{i1} - b_j) - b_1 = 6 f(10) - 1 = 5 \\
e_1 &= \frac{1}{2}\sum_{k=1}^{o}(t_{k1} - y_{k1})^2 = \frac{1}{2}(2-5)^2 = 4.5 \\
E &= \frac{1}{N}\sum_{s=1}^{N} e_s = \frac{1}{4}(e_1 + e_2 + e_3 + e_4) = \frac{1}{4}(4.5 + 0.5 + 0.5 + 4.5) = 2.5
\end{aligned}
\tag{14}
$$

The fitness value $E$ of $P_{current}$ which is the position $X_1$ is 2.5 by form (14). If $P_1$ and $GP$ are equal to $P_{current}$ , the fitness of $P_1$ and $GP$ are both 2.5. The velocity and position are updated by using form (11) - (13):

$$
\begin{aligned}
V_{11}(2) &= W(1)V_{11}(1) + C_1 r_1 (P_{11} - X_{11}) + C_2 r_2 (GP_1 - X_{11}) = 0.883*0.1 + 2*0.5*(1-1) + 2*0.5*(1-1) = 0.0883 \\
X_{11}(2) &= X_{11}(1) + V_{11}(2) = 1 + 0.0883 = 1.0883
\end{aligned}
\tag{15}
$$

The weight $w_{11} = 1.0883$ calculated by form (15) is from the first neuron $Z_1$ of input layer to the first neuron $H_1$ of hidden layer. $Z_i = Z_1, Z_2, ..., Z_6$ is the neuron of input layer and $H_j = H_1, H_2, ..., H_6$ is the neuron of hidden layer. Based on the computational process in form (15), all new weight and bias in hidden layer and output layer are 1.0883.

Finally, similarly other particles are calculated to find the $P_q$ and $GP$ by form (14) and (15) .The procedure is iteratively computed form updating $X_1$ to updating $X_{20}$ until $k = K$ or $E \le \varepsilon$ . The global best position of the optimal solution is in the last iteration. The weight and bias, which are the elements of global best position, are used to complete the network. Sample $\alpha_5$ is used to test the performance of network. The best weight and bias with 30 iteration times are shown in Table 4.

Table 4: The best weight and bias

| | | | | $w_{ji}$ | | | $b_j$ |
|---|---|---|---|---|---|---|---|
| Hidden layer | -0.7663 | 0.2033 | 0.2090 | -0.2492 | -0.0820 | -0.3592 | -0.5316 |
| | -0.4580 | -0.1974 | 0.0626 | 0.1622 | -0.0820 | -0.1621 | -0.2961 |
| | 0.6951 | -0.0029 | -0.5414 | 0.7384 | 0.3857 | -0.2092 | -0.5713 |
| | -0.2625 | 0.2793 | -0.5250 | 0.2964 | -0.6383 | -0.5118 | 0.3433 |
| | 0.1294 | 0.3717 | 0.5775 | 0.7354 | -0.3442 | 0.1935 | 0.2269 |
| | -0.0508 | 0.1956 | -0.7650 | -0.4062 | 0.5568 | 0.4017 | 0.1613 |
| | | | | $w_{kj}$ | | | $b_k$ |
| Output layer | -0.5831 | -0.5840 | 0.3394 | -0.5024 | -0.6071 | 0.2455 | -0.0703 |

Table 5 shows the mean square error and error rate of the test results by PSONN and PCNP-PSONN, and the results are computed by the average of 10 test times. Fig 2 shows the curves of prediction results by PSONN and PCNP-PSONN. Due to the factor selection by PCNP, the input variables of factor $B_1, B_2, B_3, B_4, B_7, B_8$ are selected to train the PSONN and it improves the accuracy for the single PSONN which uses input variables of factors $B_1, \ldots, B_8$, without selection. For forecasting results in this case, PCNP-PSONN is therefore more accurate than PSONN in due to factors screening process of PCNP.

Table 5: The result MSE and error rate of PCNP-PSONN and PSONN

| | Mean Square Error | Error rate |
|---|---|---|
| PSONN | 0.1070 | 26.95% |
| PCNP-PSONN | 0.0222 | 14.49% |

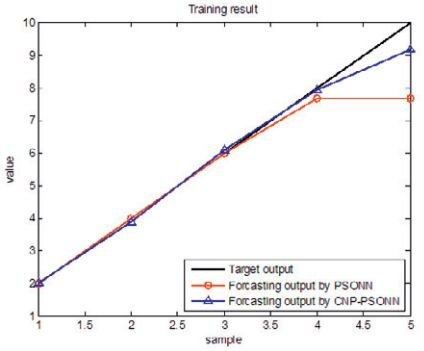


Fig.2 Target output and result output by PSONN and PCNP-PSONN

## 4. Conclusions

This paper proposes a hybrid approach of PCNP-PSONN for forecasting. An instance case demonstrates the usability and validity of this hybrid approach. This research will further investigate in two directions: further improving PSONN method and excavating the potential of combining PCNP into the other kinds of ANN.

## References

[1] A. E. Bryson, Y. C. Ho. Applied optimal control: optimization, estimation, and control. Blaisdell Publishing Company or Xerox College Publishing. pp. 481 (1969)

[2] C. K. Zhang, H. Shao, Yu Li, Particle Swarm Optimisation for Evolving Artificial Neural Network,IEEE 0-7803-6583-6/00 (2000)

[3] Ergezinger,S., Tomsen,E., An accelerated algorithm for multiplayer perceptions optimization layer by layer, IEEE Transactions Neural Networks, 2001(6), pp.31-42.

[4] F. van den Bergh, A. P. Engelbrecht, Training product unit networks usingcooperative particle swarm optimizers. Proc. Of the third Genetic and Evolutionary Computation Conference, pp. 84-90 (2001).

[5] G. M. Zhang, C. L. Qiu, X. D. Li and W. Zhu, The Risk Assessment Model of Special Equipment Based on F-AHP and ANN, IEEE Fourth International Conference on Natural Computation, 978-0-7695-3304-9/08 (2008).

[6] J. R. Zhang, J Zhang, Tat-Ming Lok, Michael R. Lyu, A hybrid particle swarm optimization – back-propagation algorithm for feedforward neural network training, Applied Mathematics and Computation 185 (2007) 1026 – 1037.

[7] K. K. F. Yuen, The pairwise opposite matrix and its cognitive prioritization operators: The ideal alternatives of the pairwise reciprocal matrix and analytic prioritization operators, J. Oper. Res. Soc., Accepted for publication (2011).

[8] K. K. F. Yuen, Cognitive network process with fuzzy soft computing technique for collective decision aiding, The Hong Kong Polytechnic University, PhD thesis (2009).

[9] K. K. F. Yuen, The primitive cognitive network process: comparisons with the analytic hierarchy process, International Journal of Information Technology & Decision Making Vol. 10, No. 4 (2011) 659–680

[10] P. Z. Tang, Z. C. Xi, The Research on BP Neural Network Model Based on Guaranteed Convergence Particle Swarm Optimization, in Second International Symposium on Intelligent Information Technology Application. (2008). pp. 13-16.

[11] R. C. Eberhart, J. Kennedy, Particle swarm optimization, in: Proc. of IEEE Int. Conf. on Neural Network, Perth, Australia (1995) 1942 – 1948.

[12] R. Poli, J. Kennedy,T. Blackwell,Particle swarm optimization, Swarm Intell (2007) 1: 33–57

[13] T. L. Saaty, The Analytic Hierarchy Process, New York: McGraw-Hill, 1980.

[14] W. Wu, G. R. Feng, Z. X. Li, and Y. S. Xu, Deterministic convergence of an online gradient method for BP neural networks, IEEE Trans. Neural Networks, vol. 16, May, 2005, pp.533-540, doi: 10.1109/TNN.2005.844903.

[15] W. Pedrycz, B.J. Park, N.J. Pizzi, Identifying core sets of discriminatory features using particle swarm optimization, Expert Systems with Applications 36 (2009) 4610–4616.